Лекция 2. СИСТЕМА ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ Ns-2

Введение (с. 2)

В настоящее время наблюдается активное развитие как сетей связи, так и услуг, предоставляемых этими сетями. Этот процесс требует не только разработки нового технологического оборудования, программных продуктов и стандартов, но и подготовки квалифицированных специалистов. Компьютерное моделирование, как показала практика, играет существенную роль при решении как тех, так и других задач. В процессе разработки модель, аппроксимирующая свойства и поведение исследуемой сети, позволяет решать задачи по оптимизации и управлению. Апробация тех или иных решений на модели несравнимо дешевле, чем на реальной системе, и позволяет исключить возможные ошибки. В учебном процессе моделирование позволяет наглядно проиллюстрировать основные принципы и дать обучаемым навыки работы с такими системами, которые физически им недоступны.

Статическое и динамическое (имитационное) моделирование (с. 3)

При моделировании сетей связи, как правило, используется один из двух подходов. Первый из них основан на теории телетрафика, которая является приложением теории систем массового обслуживания к сетям связи и базируется на принципах теории вероятности. В рамках этого подхода могут быть получены строгие математические результаты, представляемые на вероятностном языке. Недостатком подхода является сложность расчетов, делающая трудным или невозможным исследование сложных систем, особенно если они нестационарны во времени. Этот подход иногда называют статическим моделированием.

Второй подход заключается в непосредственном моделировании процессов, протекающих в системе (например, отправка и получение пакетов), с помощью специальной программы. Такой способ называют имитационным (иначе – динамическим) моделированием. Он относится к классу методов Монте-Карло в том смысле, что все случайности, имеющие место в реальной системе (случайные задержки, ошибки передачи и т. п.), моделируются с помощью датчиков случайных чисел. Каждый запуск такой модели, таким образом, является реализацией некоторого случайного процесса. Поэтому результаты моделирования в таких системах могут различаться от запуска к запуску, и для получения достоверных сведений требуется статистическая обработка результатов достаточно большого числа испытаний.

Существующее ПО для моделирования сетей связи (с. 4-5)

В настоящее время существует несколько коммерческих программных продуктов, предназначенных для имитационного моделирования сетей связи. Наиболее распространенные многофункциональные пакеты представлены в таблице.

Производитель и название	Цена	Требования к памяти компьютера	Операционные системы	Примечания
Caci Products Co. COMNET III	$35.000	от 32 МБ ОЗУ от 100 МБ HDD	Win 98/NT/2000 SunOS, Solaris	LANs, X.25, ATM, Frame Relay, протоколы маршрутизации IP. Реализация собственного кода на SIMSCRIPT. Анимация.
Cadence Inc. BONeS DESIGNER	$20.000	от 32 МБ ОЗУ от 80 МБ HDD	SunOS, Solaris, HP-UX	LANs, X.25, ATM, Frame Relay, реализация собственного кода на C++. Анимация.
MIL3 Inc. OPNET Modeler	$40.000	от 16 МБ ОЗУ от 150 МБ HDD	Win 98/NT/2000, Solaris, HP-UX	Fixed/wireless LANs, X.25, ATM, Frame Relay, Intelligent Networks, Web caching, http и т.п., реализация собственного кода на C++. Анимация. Исходный код библиотек частично открыт.
VINT project network simulator version 2 (ns2)	-	от 8 МБ ОЗУ до 250 МБ HDD	Win 95/98/ME/NT/2000, Solaris, SunOS, Linux,FreeBSD HP-UX	Fixed/wireless LANs, X.25, ATM, Frame Relay, Web caching, http, все разновидности tcp и т.п., реализация собственного кода на C++ и tcl/otcl. Анимация. Исходный код полностью открыт.

Их конкурентом является система ns-2, бесплатно распространяемая в открытых кодах на условиях лицензии GPL (GNU Public License). Соответственно, бесплатны и доступны on-line все обновления и дополнения системы (новые библиотеки, реализация новых протоколов, bug-fix и т.п.). Возможно также изменение и дополнение системы пользователем для нужд конкретной задачи (например, реализации новых протоколов для изучения их работы и сравнения их с существующими).

Полные версии, включающие все функции, доступны на многих платформах, включая SunOS, Solaris, Linux, FreeBSD, Windows 95/98/ ME/NT/2000.

Система ns-2: история (с. 6)

Основой для создания ns-2 послужил программный продукт для моделирования сетей network simulator, разрабатывавшийся в Калифорнийском университете с 1989 года и известный также под названием REAL. Работа над созданием ns-2 началась в 1996 году в рамках проекта VINT (Virtual InterNetwork Testbed) по инициативе DARPA (Defense Advanced Research Project Agency). Проект выполнялся под руководством ряда научных центров: USC/ISI (University of Southern California / Information Sciences Institute), Xerox PARC, LBNL (Lawrence Berkley National Laboratory) и UCB (UC Berkley). В настоящее время разработка ns-2 ведется в рамках проектов SAMAN (Simulation Augmented by Measurment and Analysis for Networks), финансируемого DARPA, и CONSER (Collaborative Simulation for Education and Research), финансируемого через NSF (National Science Foundation). В работе участвуют также исследователи из других организаций, в том числе из некоммерческого исследовательского института ICIR (The ICSI Center for Internet Research).

Основные возможности ns-2 (с. 7-8)

Симулятор ns-2 осуществляет имитационное моделирование сетей на уровне пакетов, то есть, моделирует генерацию пакетов и прохождение их по сети. На прикладном уровне моделируется характер трафика, порождаемого различными приложениями: Web, FTP, Telnet, RealAudio; кроме того, имеются абстрактные модели трафика, например Constant Bitrate. Возможно моделирование работы протоколов транспортного уровня UDP и различных реализаций TCP, multicast-протоколов, различных протоколов маршрутизации в проводных и беспроводных сетях, очередей с дисциплинами обслуживания DropTail и RED. Кроме того, моделируются некоторые факторы, относящиеся к физическому уровню: задержка пакетов в каналах, возникновение ошибок, видимость/невидимость узлов в беспроводных сетях (как наземных, так и спутниковых), расход энергии батарей в устройствах с автономным питанием.

Результатом работы симулятора являются выходные текстовые файлы, в которых регистрируется ход моделирования (моменты генерации/получения пакетов, состояние очередей, отброс пакетов в очередях и т. д.). Кроме того, в модель могут быть включены инструкции, вычисляющие любые величины, измерение которых требуется в конкретной задаче (задержка пакетов, пропускная способность и т. п.). Значения этих величин в ходе моделирования также могут регистрироваться в выходных файлах.

Для визуализации результатов служат аниматор NAM (Network Animator) и построитель графиков Xgraph. Кроме того, система содержит генератор топологий, упрощающий описание топологии больших сетей.

Архитектура ns-2 (с. 9)

Симулятор ns-2 состоит из двух частей. Одна из них написана на языке C++ и должна быть перекомпилирована в случае внесения изменений и дополнений; другая написана на интерпретируемом языке OTcl (объектно-ориентированное расширение языка сценариев Tcl) и, соответственно, не требует компиляции. При этом иерархии классов в обеих частях имеют совпадающие части и в терминологии ns-2 называются компилируемой и интерпретируемой иерархиями, соответственно. Взаимодействие между частями, написанными на таких принципиально разных языках программирования, осуществляется согласно спецификации, определяющей способ обращения из tcl-сценария (скрипта) к любому методу классов компилируемой иерархии и возвращение назад результатов, а также способ обращения из программы на C++ к любому методу, описанному в Tcl-сценарии. Во втором случае, фактически, интерпретатор языка Tcl вызывается из C++ как функция. По замыслу создателей ns-2, все методы, имеющие дело с отдельными пакетами и потому требующие высокого быстродействия, относятся к компилируемой части. Интерпретируемая же часть отвечает за менее частые события, чем передача пакетов, обеспечивающие управление ходом моделирования, и манипуляцию объектами, описанными в компилируемой части. Также, Tcl-скриптом является собственно описание модели сети, подлежащей исследованию.

Такой подход позволяет быстро построить требуемую модель сети с помощью скриптового языка OTcl без необходимости вникать в структуру компилируемой части ns-2. В случае, если необходима модификация или дополнение компилируемой части, это может быть сделано путем добавления (или изменения) C++ кода и перекомпиляции системы. Единственный недостаток такого подхода – трудности при изучении системы и отладке программ (моделей), возникающие вследствие использования двух языков.

Основные понятия (с. 10)

С точки зрения объектного подхода в имитационном моделировании, принятого в ns-2, сеть представляет собой совокупность сетевых объектов (Network Objects), каждый из которых способен определенным образом реагировать на некоторое множество событий (Events). Обо всех событиях сетевые объекты извещаются так называемым планировщиком (Scheduler), который содержит хронологическую таблицу событий. Каждое событие имеет обязательные атрибуты: время наступления и сетевой объект, к которому относится это событие. В системе ns-2 пакет – это событие, сгенерированное в объекте-получателе другим сетевым объектом (отправителем). Так как обмен пакетами – это единственный способ взаимодействия между сетевыми объектами, пакет – это единственный тип «внутреннего» события в системе. Все остальные типы событий генерируются пользователем при описании модели. Они называются at-событиями. Такие события предназначены для управления ходом моделирования (включение/выключение генератора трафика, обрыв/восстановление линии связи и т.п.).

Планировщик событий (с. 11)

Планировщик по очереди (в хронологическом порядке) выбирает из таблицы события и извещает соответствующие сетевые объекты о наступлении этих событий, сообщая также момент времени, в который произошло это событие. С точки зрения программиста, такое «извещение о событии» представляет собой просто вызов метода-обработчика для соответствующего

объекта. Вся сопутствующая информация передается через параметры этого метода.

Помещать события в таблицу планировщика могут как сами сетевые объекты, так и пользователь при описании модели. Как говорилось выше, в первом случае событие называется пакетом, во втором – at-событием.

Таким образом, любое взаимодействие между сетевыми объектами осуществляется через посредство планировщика. Отправка пакета одним сетевым объектом другому есть ни что иное, как постановка отправителем в таблицу события, адресованного получателю и помеченного тем моментом времени, когда пакет должен до этого получателя дойти (возможно, с учетом задержки при передаче). Получение пакета, соответственно, представляет собой извещение планировщиком получателя о событии и обработка этого события получателем.

При этом (если речь не идет о планировщике реального времени) модельное время не имеет никакого отношения к реальному. Планировщик извлекает события из таблицы, обращая внимание лишь на порядок их следования. Интервал между обработкой последовательных событий определяется лишь скоростью их обработки компьютером и не зависит от того, какими моментами модельного времени помечены эти события в таблице. Модельное время находит свое отражение лишь в трассировочных файлах, куда сетевые объекты записывают моменты отправки и получения пакетов (естественно, в модельном времени).

В системе ns-2 предусмотрен также планировщик реального времени (Real-Time Scheduler), который синхронизует модельное время с реальным, ожидая между выборкой последовательных событий из таблицы по возможности в точности столько времени, сколько составляет разность между моментами, которыми помечены эти события. Такие планировщики используются, когда компьютерная модель входит в состав реальной сети. Например, создав на одном или нескольких компьютерах модель большой сети с планировщиком реального времени и подключив к такой модели два других компьютера, можно реалистично имитировать ситуацию, когда эти компьютеры связываются друг с другом, находясь в любых двух заданных точках этой большой сети. Этот планировщик, однако, далее в данной лекции не рассматривается.

Структура пакета (с. 12)

Из соображений простоты все пакеты в системе имеют одинаковый формат (являются объектами одного класса), и поэтому каждый пакет, независимо от того к какому протоколу он относится, имеет в своем составе заголовки *всех* протоколов, используемых в модели. Типичная структура пакета в ns-2 изображена на рис. 1. Поле данных используется лишь в случаях, когда ns2-модель входит в состав реальной сети; если модель чисто компьютерная, то достаточно иметь в заголовке поле, содержащее размер пакета. Для ускорения процесса моделирования необходимо указывать явным образом, какие протоколы будут использоваться в модели, чтобы исключить заголовки неиспользуемых протоколов из заголовка пакета.

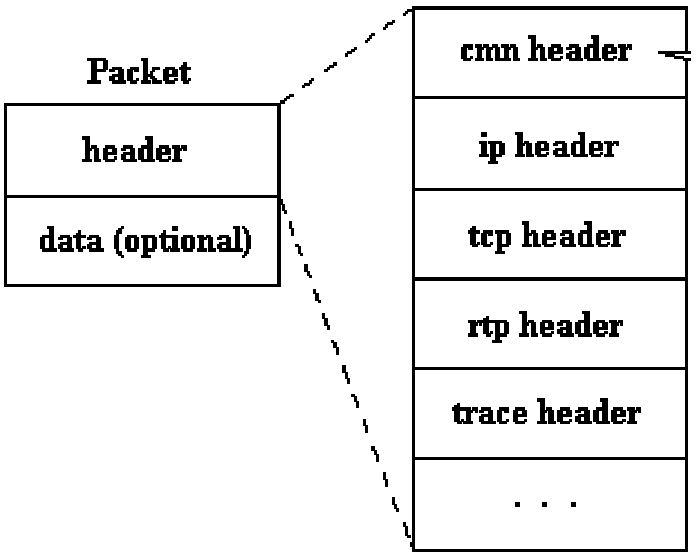

Рис. 1 Структура пакета

Сетевые объекты: простые и составные (с. 13)

Сетевые объекты в ns-2 могут быть простыми и составными. К простым объектам относятся очереди, линии задержки, мультиплексоры, демультиплексоры, а также такие нетривиальные объекты, как агенты и приложения (см. далее). Составные объекты формируются путем объединения простых объектов. Система ns-2 содержит большой набор готовых описаний классов составных объектов, среди которых узлы (Nodes), линии связи (Links), локальные сети (как проводные 802.3 Ethernet, так и беспроводные 802.11) и даже спутниковые сети (как на геостационарных, так и на низкоорбитальных спутниках). Однако, система может быть расширена введением в нее новых классов сетевых объектов. Особенно удобно создавать составные объекты на основе имеющихся (как простых, так и составных). При этом для описания новых классов может использоваться либо язык C++, либо OTcl. В случаях, когда важна производительность, используется C++ и перекомпилируется вся система; наоборот, когда производительность некритична и на первое место выходит легкость и быстрота разработки, используется OTcl, который вообще не требует компиляции.

Абстракция сетевого уровня: узел (с. 14)

Абстракцией сетевого уровня в системе ns-2 является узел. Узел может принимать пакеты и классифицировать их по адресу и порту, являющимся полями IP-заголовка. Как было отмечено выше, узел является составным объектом. Простейшая структура узла изображена на рис. 2. Прибывающие в узел пакеты принимаются входной точкой (Entry Point) узла и классифицируются классификатором адреса (Address Classifier), представляющим собой демультиплексор. Пакеты, адресованные другим узлам, передаются на одну из линий связи, подключенных к узлу, в соответствии с маршрутной таблицей, принадлежащей классификатору адреса. Пакеты же, адресованные данному узлу, передаются классификатору порта (Port Classifier), также являющемуся демультиплексором. В зависимости от номера порта, классификатор передает пакет одному из прикрепленных к узлу агентов (Agents), которые являются сетевыми объектами, не входящими в состав узла и

отвечающими за реализацию протоколов более высокого уровня (например, транспортных протоколов UDP и различных реализаций TCP, протоколов маршрутизации и пр.). Отметим, что узел может иметь практически любое количество связей, а также портов с прикрепленными к ним агентами. Существуют также более сложные multicast-узлы, способные рассылать прибывающие пакеты нескольким адресатам, но они в данной работе не используются.

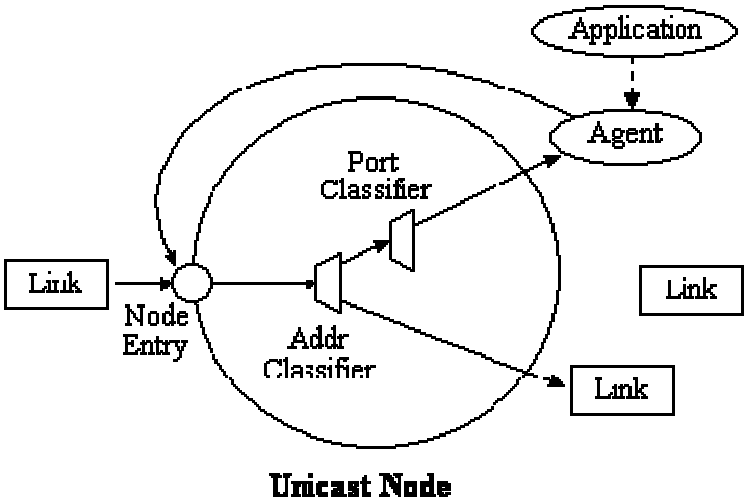

Рис. 2 Структура узла

Маршрутная таблица узла может быть как статической, так и динамической. Во втором случае она формируется агентом маршрутизации, подключенным к узлу и получающим информацию о топологии сети путем обмена специальными пакетами с аналогичными агентами на других узлах (как это и делается при динамической маршрутизации в реальных сетях).

Абстракции транспортного и прикладного уровней: Агенты и Приложения (с. 15)

Для создания TCP-соединения необходимо создать два агента: TCP-передатчик и TCP-приемник. TCP-приемник лишь принимает пакеты и посылает назад уведомления о получении (ACK). Поэтому такой агент в системе один: TCPSink. Передатчик же может вести себя различным образом в зависимости от порядка и моментов получения им уведомлений от адресата о получении отправленных им пакетов. Соответственно, в системе ns-2 имеется несколько вариантов TCP-передатчика, моделирующих различные реализации протокола TCP (TCP, TCP/Tahoe, TCP/Reno, TCP/Vegas), использующие разные комбинации алгоритмов TCP (Slow Start, Congestion Avoidance, Fast Recovery, Fast Retransmit).

С протоколом UDP все обстоит гораздо проще, потому что уведомления о получении там не предусмотрены. Для передачи используется агент UDP, а для приема – агент Null, просто принимающий и отбрасывающий все приходящие пакеты.

За прикладной уровень в ns-2 отвечают приложения (Applications). Приложение прикрепляется к агенту и служит для создания трафика. В ns-2 имеются приложения, моделирующие трафик, характерный для реальных протоколов прикладного уровня (FTP и Telnet), а также абстрактные генераторы трафика различного типа (например, CBR – простейший генератор трафика с

постоянным темпом выдачи пакетов). Приложения запускаются и останавливаются пользовательскими at-событиями.

Линии связи (с. 16)

Дуплексные (двусторонние) линии связи в ns-2 являются составными объектами, состоящими из двух противоположно направленных симплексных (односторонних) линий связи.

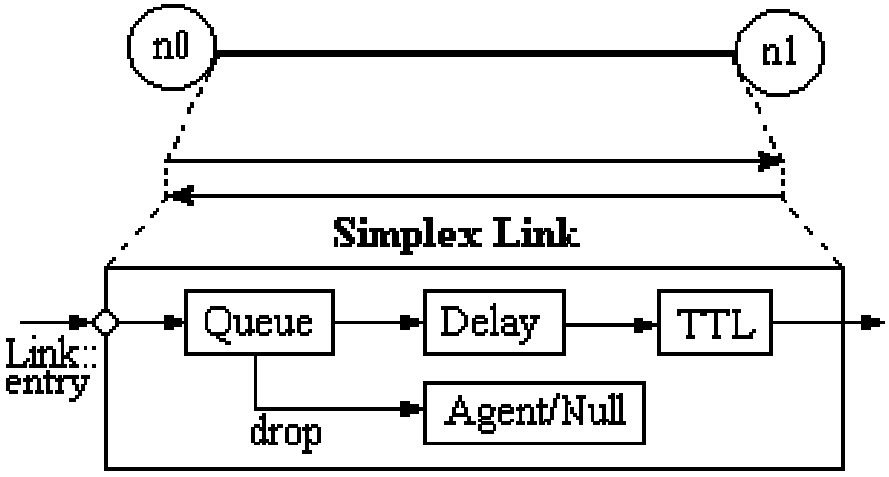

Рис. 3 Структура линии связи

Симплексная линия связи также является составным объектом (см. рис. 3). Она состоит из очереди, Null-агента, в который отправляются отброшенные из очереди пакеты, линии задержки и объекта, проверяющего и уменьшающего на единицу поле времени жизни (Time to Live, TTL) пакета. Кроме того, могут быть еще объекты-мониторы, следящие за состоянием очереди (постановка в очередь, извлечение, отброс пакетов). Заметим, что все эти объекты, за исключением линии задержки, физически относятся скорее к узлу, чем к линии связи. Объяснить, почему создатели ns-2 ввели их все же в состав линии связи, можно двумя способами.

Во-первых, очередь должна «знать», с какой скоростью ей выдавать пакеты в линию связи, а эта скорость в модели определяет пропускную способность линии связи (хотя в реальности, наоборот, скорость передачи пакетов определяет*ся* пропускной способностью линии). Кроме очереди, нет ни одного другого объекта, через параметры которого можно было бы ввести в модель пропускную способность линии. Поскольку пропускная способность является атрибутом именно линии связи, то логично, чтобы очередь (а значит, и все вспомогательные объекты, которые к ней подсоединены), входила в состав линии связи.

Во-вторых, такой подход позволил более равномерно распределить сложность по сетевым объектам. Иначе от линии связи осталась бы лишь линия задержки, а узел бы существенно усложнился.

Пример: соединение двух узлов по протоколу TCP (с. 17)

Рассмотрим процессы, происходящие в простейшей модели, состоящей из двух узлов n0 и n1, соединенных дуплексной линией связи, к первому из которых подключен агент-передатчик TCP с генератором трафика FTP, а ко второму – агент-приемник TCPSink.

Агент TCP, управляемый генератором трафика FTP, генерирует пакет и передает его на входную точку «своего» узла n0. Внутри узла пакет попадает на

классификатор адреса. Поскольку пакет адресован узлу n1, классификатор передает его на исходящую линию связи. Линия связи, прибавляя ко времени возникновения пакета соответствующее время задержки, передает его на входную точку узла n2.

Внутри узла n2 пакет через классификатор адреса и классификатор порта (имеющий единственный порт, т.к. к узлу подключен один агент) попадает на агент-приемник TCPSink. Агент отправляет назад пакет-уведомление (ACK), которое попадает к агенту-передатчику TCP на узле n0 способом, аналогичным описанному выше. После этого передается новый TCP-пакет, и т. д.

Пример модели в ns-2 (с. 18 – 30)

(с. 18) Рассмотрим стандартный простой пример построения модели в ns-2. Система включает четыре узла n0, n1, n2, n3. Узел n2 соединен с узлами n0 и n1 линиями связи с пропускной способностью 2 Мбит/с и временем задержки 10 мс. Тот же узел n2 соединен с узлом n3 линией с пропускной способностью 1,7 Мбит/с и задержкой 20 мс. На узле n0 располагается агент TCP с генератором трафика FTP, на узле n1 – агент UDP с генератором трафика CBR (constant bitrate), создающий постоянный трафик 1 Мбит/с с размером пакетов 1 Кб. К узлу n3 подключен агент-приемник TCPSink и агент Null для приема UDP пакетов.

Далее рассмотрим структуру Otcl-скрипта, описывающего модель и запускающего симулятор.

(с. 19)

Создадим объект-симулятор ns и зададим цвета для отображения потоков данных в визуализаторе NAM:

```
#Create a simulator object
set ns [new Simulator]

#Define different colors for data flows (for NAM)
$ns color 1 Blue
$ns color 2 Red
```

Откроем файл записи трассировки моделирования

```
#Open the NAM trace file
set nf [open out.nam w]
```

Зададим симулятору опцию трассировки для последующей визуализации в NAM и укажем файл для записи:

```
$ns namtrace-all $nf
```

(с. 20)

Определим процедуру завершения, в которой очищается буфер трассировки, закрывается выходной файл и запускается визуализатор NAM:

```
#Define a 'finish' procedure
proc finish {} {
        global ns nf
        $ns flush-trace
        #Close the NAM trace file
        close $nf
        #Execute NAM on the trace file
```

```
        exec nam out.nam &
        exit 0
}
```

(с. 21)
Создадим четыре узла:

```
#Create four nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
```

Создадим линии связи с заданными параметрами и дисциплиной обслуживания DropTail (отброс приходящих пакетов, если очередь переполнена). Зададим максимальный размер очереди (объем буфера) для линии связи между n2 и n3.

```
#Create links between the nodes
$ns duplex-link $n0 $n2 2Mb 10ms DropTail
$ns duplex-link $n1 $n2 2Mb 10ms DropTail
$ns duplex-link $n2 $n3 1.7Mb 20ms DropTail
#Set Queue Size of link (n2-n3) to 10
$ns queue-limit $n2 $n3 10
```

(с. 22)
Зададим служебные параметры для визуализатора NAM: положение связей на отображаемой схеме и отображение состояния линии связи n2-n3

```
#Give node position (for NAM)
$ns duplex-link-op $n0 $n2 orient right-down
$ns duplex-link-op $n1 $n2 orient right-up
$ns duplex-link-op $n2 $n3 orient right

#Monitor the queue for link (n2-n3). (for NAM)
$ns duplex-link-op $n2 $n3 queuePos 0.5
```

(с. 23)
Создадим агент TCP и подключим его к узлу n0:

```
#Setup a TCP connection
set tcp [new Agent/TCP]
$tcp set class_ 2
$ns attach-agent $n0 $tcp
```

Создадим агент TCPSink и подключим его к узлу n3:

```
set sink [new Agent/TCPSink]
$ns attach-agent $n3 $sink
```

Укажем, что агент TCP должен передавать данные агенту TCPSink:

```
$ns connect $tcp $sink
$tcp set fid_ 1
```

(с. 24)
Создадим агент UDP и подключим его к узлу n1:

```
#Setup a UDP connection
```

```
set udp [new Agent/UDP]
$ns attach-agent $n1 $udp
```

Создадим агент Null и подключим его к узлу n3:

```
set null [new Agent/Null]
$ns attach-agent $n3 $null
```

Укажем, что агент UDP должен передавать данные агенту Null

```
$ns connect $udp $null
$udp set fid_ 2
```

(с. 25)
Создадим генератор трафика FTP и подключим его к агенту TCP

```
#Setup a FTP over TCP connection
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ftp set type_ FTP
```

Создадим генератор трафика CBR, подключим его к агенту UDP и зададим параметры

```
#Setup a CBR over UDP connection
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set type_ CBR
$cbr set packet_size_ 1000
$cbr set rate_ 1mb
$cbr set random_ false
```

(с. 26)
Зададим at-события, управляющие ходом моделирования:
- пуск и остановка генераторов трафика:

```
#Schedule events for the CBR and FTP agents
$ns at 0.1 "$cbr start"
$ns at 1.0 "$ftp start"
$ns at 4.0 "$ftp stop"
$ns at 4.5 "$cbr stop"
```

- отключение агентов TCP TCPSink от соответствующих узлов (приводим для примера; в данном случае в этом нет необходимости, т.к. моделирование на этом заканчивается)

```
#Detach tcp and sink agents (not really necessary)
$ns at 4.5 "$ns detach-agent $n0 $tcp; $ns detach-
agent $n3 $sink"
```

- вызов определенной ранее процедуры finish после 5 секунд модельного времени

```
#Call the finish procedure after 5 s of sim. time
$ns at 5.0 "finish"
```

(с. 27)
Запустим моделирование

```
#Run the simulation
$ns run
```

Заключение

В данной лекции были рассмотрены основы имитационного моделирования сетей связи на примере известной системы с открытым кодом ns-2. Были изложены принципы имитационного моделирования, основы архитектуры ns-2, основы реализации объектного подхода к моделированию в ns-2. Кроме того, был разобран стандартный пример построения модели в ns-2. Подробные сведения об ns-2 и примеры применения этой системы можно получить по указанным ниже интернет-адресам.

Интернет-ресурсы

1. Сайт разработчиков http://www.isi.edu/nsnam/ns/
    a. официальное руководство http://www.isi.edu/nsnam/ns/ns-documentation.html
    b. коллекция NS2-моделей http://www.isi.edu/nsnam/repository/
2. Практическое руководство http://nile.wpi.edu/NS/
3. Пособие по TCP с использованием NS2 http://www.ce.chalmers.se/~fcela/tcp-tour.html
4. Русскоязычный сайт http://www.cs.tut.fi/~yk/ns2ru/